The Thesis Committee for Corey Leonard Marcus
certifies that this is the approved version of the following thesis:

Direct Monocular SLAM Augmented with LIDAR Range Measurements

SUPERVISING COMMITTEE:

Renato Zanetti, Supervisor

Maruthi Akella

Direct Monocular SLAM Augmented with LIDAR Range Measurements

by

Corey Leonard Marcus

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2020

Dedicated to all those who have stopped to help along the way.

"If I have seen further it is by standing on the shoulders of giants."

– Sir Isaac Newton

Acknowledgments

I'd first and foremost like to acknowledge my adviser Dr. Renato Zanetti for the guidance and expertise provided throughout the making of this thesis. In addition, I'd like to thank the friends, familiy, colleagues, and loved ones who are too numerous to count. Their support and encouragement has been an invaluable resource. Finally, I'd like to extend my gratitude to NASA. I am honored to have been selected as a research fellow under NASA's NSTGRO Program.

This work was supported by a NASA Space Technology Graduate Research Opportunity.

Direct Monocular SLAM Augmented with LIDAR Range Measurements

Corey Leonard Marcus, M.S.E.
The University of Texas at Austin, 2020

Supervisor: Renato Zanetti

There are many situations where a spacecraft requires knowledge of the geometry and pose relative to the surrounding environment. SLAM is a class of algorithms capable of solving these problems simultaneously. In this thesis we present a Direct Monocular SLAM system which is augmented with flash LIDAR images. LIDAR measurements provide three tangible improvements to Direct Monocular SLAM. Improved SLAM system initialization through the use of LIDAR. Metric LIDAR measurements allow the true scale of localization and mapping estimates to become observable. And finally, an Extended Kalman filter is used to provide updates on map features using LIDAR images. Monte Carlo methods are used to demonstrate that incorporating LIDAR measurements into the system provides significant performance improvements over a system without LIDAR.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Simultaneous Localization and Mapping

Every time we enter a new room our brains perform two tasks simultaneously; they construct a map of the room's geometry, and they provide a location of our body within that geometry. These are critical steps in allowing us to control our motion or interact with our environment. There is a large interdisciplinary effort to create algorithms which allow computers to solve these mapping and localization problems.

With a simple thought experiment it becomes clear that mapping and localization are intrinsically coupled problems. Imagine you sit in a car moving 100 feet per second with your eyes closed. You then open and close your eyes twice with a one second interval. In the first moment you notice yourself passing a red house and in the second a blue one. It might seem logical to conclude that these two houses are 100 feet apart. However, your true speed can never be known with absolute certainty. This uncertainty induces error into your distance calculation. Thus your knowledge of speed (localization) and distance between the houses (mapping) are inherently linked. Acknowledging the correlation between these two problems and solving them simultaneously

allows for more accurate estimates of both. This is known as Simultaneous Localization and Mapping (SLAM) [1].

SLAM can provide a solution for a large number of real world problems. These include situations such as autonomous driving [2], infrastructure inspection [3], office supply inventory management [4], and autonomous landing [5]. The sensors which can be used for SLAM are equally numerous. SLAM algorithms have been created using almost any combination of visual [6], LIDAR [7], intertial [8], depth [9], radar [10], sonar measurements [11], *etc.* Most common among these are visual sensors, *i.e.* cameras. Monocular visual SLAM's primary drawback is that the scale of the scene is not observable. In other words, the system is not able to distinguish between small movements in a small environment and large movements in a large one.

1.2 Motivation

One particular case of interest in the Aerospace field is situations involving free-fall. In these situations, accelerometers only provide information when the vehicle produces thrust. A well designed SLAM system should be robust and continue to function without vehicle thrust, thus we assume we cannot rely on acceleration measurements. To further improve robustness, we assume gyroscopic measurements are unavailable as well. With these assumptions, our system becomes independent of all inertial measurements. Two examples of free-fall situations using SLAM are on-orbit proximity operations and autonomous landing. In proximity operations, one spacecraft may desire

to inspect, maintain formation, or dock with another. In autonomous landing, a spacecraft may desire to create a map of the environment below it in order to search for landing sites. It is clear that both of these problems could be solved with SLAM.

The metric scale of the scene must be estimated in order to provide a meaningful solution. Since these situations involve free-fall, inertial measurements cannot be relied upon. Light Detection and Ranging (LIDAR) provides an alternative. It is straightforward to use LIDAR for scale estimation as it provides metric measurements of ranges to map points. In addition, LIDAR has a good heritage in spaceflight, notably used for autonomous docking with the ISS on Orbital ATK's Cygnus, Figure 1.1, and SpaceX's Dragon vehicles [12]. This means that there is a strong application in spaceflight for visual plus LIDAR SLAM algorithms. This thesis develops and tests a SLAM algorithm using monocular camera and LIDAR measurements for use in spaceflight and other related applications.

Figure 1.1: The Cygnus Spacecraft during proximity operations [13].

Chapter 2

Literature Review

2.1 The Origin of SLAM

The basic premise of SLAM is to simultaneously refine estimates of the environment around the vehicle and the vehicle's location within that space. Measurements of the environment are often relative to the sensor. For example, images show the camera's 1^{st}-person point of view instead of a 3^{rd}-person one, while LIDAR measures ranges from the sensor instead of absolute positions. This causes a coupling between estimates of sensor location and environment geometry. Poor estimates of one negatively affect the other. SLAM is effective because it seeks to minimize error across mapping and localization simultaneously. First proposed in 1988 by Cheeseman et al [14], SLAM is not a new development. However, advances in computational power meant that SLAM with cameras was not implementable on a large scale or in real-time until much more recently. Klein and Murray's Parallel Tracking and Mapping (PTAM) [15] represents one of the earliest and best known fully fledged SLAM architectures.

Early SLAM techniques centered around an Extended Kalman Filter [16]. A naive implementation of this form requires inversion of an $n \times n$

matrix where n is the number of map features. Consequently, these implementations are not real-time capable in systems containing tens to hundreds of thousands of map features. These difficulties, combined with non-linearities in dynamics and observation models led to the development of an alternative filtering approach. Much work was done to provide a filtering SLAM system based on the Rao-Blackwellised Particle Filter (RBPF) [17]. One famous example of RBPF is FastSLAM [18] by Montemerlo and Thrun. FastSLAM is able to handle real-time operation with many more map features than EKF-SLAM as it has computational complexity of $\mathcal{O}(k \log(n))$ where k is the number of particles. This can be compared to the $\mathcal{O}(k^2)$ computational complexity of EKF-SLAM.

2.2 Two Competing Philosophies

The RBPF represented a paradigm shift in filtering SLAM as it allowed nonlinearities in localization to be handled with a particle filter. Each particle tracks its own map representation which is updated while conditioning on the particle's localization estimate. Once this statistical assumption is made, the location of each map feature becomes independent from the others. This reduces the computational complexity significantly when compared to EKF-SLAM as large matrix inversions are no longer required.

Simultaneously to the work on RBPF-SLAM, many researchers were working on the SLAM problem from outside the Baysiean perspective. The aforementioned PTAM is one notable example. These researchers approached

SLAM as a Bundle Adjustment [19] problem. The *bundles* of observations to each map feature are *adjusted* with a large non-linear least squares solver. These adjustments provide a SLAM estimate which minimizes an error metric such as re-projection error [20]. It was not long before researchers attempted to quantify the differences between the two SLAM perspectives. The seminal paper *Real-time monocular SLAM: Why Filter?* [21] by Strasdat *et al* argued strongly in favor of Bundle Adjustment based methodologies. The community listened and the bulk of SLAM research afterwards chose to focus on Bundle Adjustment methods.

2.3 Modern SLAM

Current bundle adjustment visual SLAM can be further divided into two approaches, feature based and direct. Feature based methods form the bulk of modern SLAM algorithms. State of the art methodologies such as ORB-SLAM2 [9] use a feature identification and matching algorithm such as ORB [22], SURF [23], or SIFT [24] to identify objects which are likely to correspond to the same physical features across multiple images. The features tend to correspond to things such as corners or changes in color on a surface. Feature based methods struggle with large textureless environments and discard the remaining information in an image after features have been matched. Direct approaches seek to align the intensities of each pixel across multiple images. Every image is used "directly" in the SLAM algorithm. These types of algorithms tend to produce much denser maps than feature based approaches,

as every pixel in an image could potentially become a map point. Until recently, direct SLAM was not real-time capable, but methodologies such as LSD-SLAM [25] have shown that it can be used to create semi-dense maps in real time with only a CPU. Direct SLAM also struggles with textureless environments, and textureless segments of the environment are often sparsely mapped.

There is a third type of SLAM which is used for some applications, occupancy map SLAM [26] [27] [28]. These algorithms do not use visual features or image pixels at all. Instead the environment is divided into a discrete domain. The SLAM system estimates if each discrete volume is occupied or empty. These types of systems are often used when the SLAM map will be used for path-planning [29].

SLAM with a monocular camera has one notable drawback, scale is not observable. Resolution of this problem necessitates additional sensors. Inertial sensors are frequently used to estimate the sensor translation between images, thus resolving scale. One such example of this utilization is shown by Mur et al [30]. LIDAR provides another convenient method for scale where measurements are used to estimate the depth of a feature [31] or pixels by Shin *et al* [32]. Shin concludes that direct SLAM is a better candidate for LIDAR integration because the association between sparse LIDAR measurements and sparse features can be tenuous. They integrate LIDAR measurements into their SLAM system through inclusion in their non-linear least squares optimization scheme. That is to say, they do not take a Bayesian approach to

updating the map with information provided by LIDAR. Furthermore, they discard a large number of LIDAR measurements which align with low texture regions of the images.

Chapter 3

SLAM Algorithm

3.1 LSD-SLAM

The work in this thesis expands upon the framework provided by Large Scale Direct SLAM (LSD-SLAM) [25] by including LIDAR images. LSD-SLAM is an open-source direct monocular SLAM algorithm. Direct SLAM algorithms work without the feature matching algorithms found in many other SLAM systems such as ORB-SLAM2 [9]. Instead, the depths of individual pixels are estimated directly. LSD-SLAM divides the global map into a series of local maps which are anchored on selected images called keyframes. LSD-SLAM chooses to estimate the *inverse* depth and associated variance for a subset of pixels in each keyframe. Because scale is unobservable for monocular camera based systems, the map produced by LSD-SLAM is non-metric. The inverse depth estimates are scaled such that their mean is one. LIDAR measurements can be used to estimate the true scale of the environment and create metric maps.

LSD-SLAM was chosen for the base of my system primarily because it is open-source and for the density of maps created. A dense map is beneficial for LIDAR measurement incorporation as it increases the likelihood that a

LIDAR measurement directly impinges upon a map point.

3.2 LSD-SLAM Improvements

LIDAR measurements have been used to change two portions of LSD-SLAM. First, a LIDAR image is used to initialize the map on system start-up. Second, LIDAR measurements are used to estimate scale and refine the map in the mapping thread. The mapping thread is outlined below in Figure 3.1.

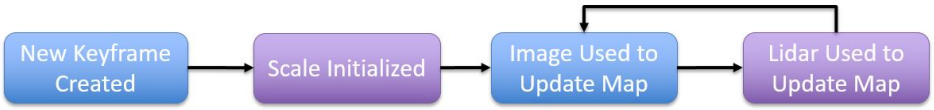

Figure 3.1: An overview of the SLAM system's mapping thread. The blue elements are found within LSD-SLAM while the purple ones have been created as part of this thesis.

The mapping thread begins when a new keyframe is initialized. Next, LIDAR measurements are used to estimate the keyframe scale. From this point until a new keyframe is created, subsequent LIDAR and RGB images are used to refine the map. The LIDAR based modifications are outlined below.

3.2.1 Map Initialization

LSD-SLAM chooses to initialize the map randomly with a large variance. This thesis presents a method which initializes map points with LIDAR measurements when possible, and randomly when no LIDAR measurement is present. For the first RGB and LIDAR image pair, every pixel matching a LIDAR measurement is initialized to a depth prescribed by that measurement.

The remainder are initialized randomly. This process is non-trivial as LIDAR measurements are of range while map elements are inverse depth and scaled to have an average estimate of one. Note that the camera and LIDAR system are assumed to be co-located, thus LIDAR measurements can always be associated with image pixels when the RGB and LIDAR images are collected at the same time.

We have a range measurement d_i which has been corrupted with zero-mean noise, ω. The variance of this noise is defined as σ_ω^2. This range measurement is matched with a pixel using the camera calibration matrix. We would like to initialize the inverse depth estimate of this pixel according to this measurement. We assume the ω is distributed according to an Inverse Gamma. The simulation outlined in the following sections actually uses an ω which is distributed according to a Gaussian with zero-mean and variance σ_ω^2, but the approximation as an Inverse Gamma is good for Gaussians with a large positive mean and small variance. These assumptions are well suited to our LIDAR system where the range measurements tend to be several orders of magnitude larger than σ_ω^2 and are always positive.

The Inverse Gamma distribution is defined by the shape, a_i, and scale, b_i, parameters. These can be found as function of d_i and σ_ω^2 in Equations (3.1) and (3.2).

$$a_i = 2 + \frac{d_i^2}{\sigma_\omega^2} \tag{3.1}$$

$$b_i = d_i \left(1 + \frac{d_i^2}{\sigma_\omega^2}\right) \tag{3.2}$$

We have a range measurement but are interested in a depth measurement, thus we must scale the Inverse Gamma distribution by a factor of $\cos(\theta_i)\cos(\phi_i)$ from Equation (3.24). When scaling an Inverse Gamma distribution, the scale parameter must also be scaled. We term the new scale parameter $\bar{b}_i$ and find it according to, $\bar{b}_i = \cos(\theta_i)\cos(\phi_i)b_i$. We have a new random variable, $x_i \sim \mathcal{G}^{-1}\left(a, \bar{b}\right)$, representing the depth of the map point.

We are interested in the distribution of the inverse depth, $\rho_i = 1/x_i$. The relationship between Gamma and Inverse Gamma distributions is known. ρ_i is distributed according to a Gamma distribution with a shape parameter, a_i, and in inverse scale parameter, $\bar{b}_i$. Thus, $\rho_i \sim \mathcal{G}\left(a, \bar{b}\right)$.

The inverse depths are all scaled such that their mean is equal to one. Thus we introduce a new scale factor, $k = \text{mean}(\frac{1}{x_i})$, and redefine $\rho_i = \frac{1}{kx_i}$. The inverse scale parameter must be multiplied by k. The new distribution is $\rho_i \sim \mathcal{G}\left(a_i, k\bar{b}_i\right)$. The mean, $\hat{\mu}_i$, and variance, $\hat{\sigma}_i^2$, of ρ_i are found according to the following:

$$\hat{\mu}_i = \frac{a_i}{k\bar{b}_i} = \frac{2 + \frac{d_i^2}{\omega^2}}{k\cos(\theta_i)\cos(\phi_i)d_i\left(1 + \frac{d_i^2}{\sigma_\omega^2}\right)} \tag{3.3}$$

$$\hat{\sigma}_i^2 = \frac{a_i}{\left(k\bar{b}_i\right)^2} = \frac{2 + \frac{d_i^2}{\omega^2}}{\left(k\cos(\theta_i)\cos(\phi_i)d_i\left(1 + \frac{d_i^2}{\sigma_\omega^2}\right)\right)^2} \tag{3.4}$$

3.2.2 Scale Initialization

When the camera has rotated and translated sufficiently far from the origin of the local map, a new keyframe is created. Points in the current map are propagated into the new one in order to initialize. The LIDAR image associated with the keyframe is used to estimate the scale between the non-metric map and the metric environment, s. This scale is defined such that a non-metric quantity y_1 could be converted to a metric representation, y_2 with $y_2 = sy_1$.

Each LIDAR range measurement, d_i, has a corresponding azimuth and elevation. This information is used to map d_i into a 3D vector, P_i^l, expressed in the frame of the LIDAR. This is demonstrated in Equation (3.24). P_i^l is matched with a pixel in the keyframe using the camera calibration matrix. A check is performed to see if this pixel has a valid inverse depth estimate ρ_i and associated variance σ_i^2. If so, these and the depth element of the 3D vector, $P_i^l(3)$, are used to form a measurement of the scale, β_i, according to Eqn. (3.5).

$$\beta_i = \rho_i P_i^l(3) \tag{3.5}$$

A maximum likelihood estimator is employed to estimate the scale, $\hat{s}$. The likelihood function is taken as a Gaussian sum, where each element of the sum is distributed according to $\mathcal{N}(\beta_i, \sigma_i^2)$. Thus, $\hat{s}$ is given according Eqn (3.6). $\hat{s}$ is found by first taking a Least-Squares estimate of s. The domain

surrounding this Least-Squares estimate is discretized and the Gaussian sum is evaluated at each discrete point. The point with the highest likelihood is accepted as $\hat{s}$.

$$\hat{s} = \arg\max_{s} \sum_{i} \mathcal{N}\left(s; \beta_i \sigma_i^2\right) \tag{3.6}$$

This method is preferred over a gradient based optimization scheme due to the large number of local maximums in the Gaussian sum.

3.2.3 Map Update With Lidar

Each new LIDAR image after the keyframe is used to update the map. The coordinate frame of the senor during measurement is termed the observer frame. Again, each LIDAR range measurement, d_i, is used to form a 3D vector, P_i^o, in the observer frame. It can be mapped into the non-metric map frame according to Equation (3.7).

$$P_i^m = \frac{1}{\hat{s}} R_{o2m} P_i^o - t_{m2o}^m \tag{3.7}$$

P_i^m can then be associated with a pixel in the keyframe using the camera calibration matrix. If this pixel has a valid inverse depth estimate, d_i will used to perform an Extended Kalman Filter (EKF) update. If it does not, an inverse depth estimate will be initialized as detailed in Section 3.2.3.1.

For the case in which an inverse depth estimate, ρ_i, already exists, it must first be mapped into a predicted measurement in the observer frame.

The mapping of $\hat{\rho}_i$ into a $\bar{d}_i$ is given by Eqn. (3.8)

$$\bar{d}_i = \hat{s}\sqrt{\frac{1}{\left(\hat{\rho}_i \hat{P}_i^m(3)\right)^2} - \frac{2}{\hat{\rho}_i \hat{P}_i^m(3)}\left(\hat{P}_i^m \cdot t_{m2o}^m\right) + (t_{m2o}^m \cdot t_{m2o}^m)} \tag{3.8}$$

Where $\hat{P}_i^m$ is the unit vector along P_i^m and $(a \cdot b)$ indicates the inner product of vectors a and b.

In practice, direct estimation of ρ_i with an EKF or UKF leads to a smug and biased filter. This is due to the non-linearity of the inverse operation. Noting that any inverse depth in the map will be greater than zero, we assume that the inverse depth has a probability density function given according to a Gamma distribution, $\mathcal{G}(a_i, b_i)$. Shape parameter a_i and inverse scale b_i can be found as analytic expressions of the distribution's mean and variance, ρ_i and σ_i^2. These are given in Equations (3.9) and (3.10).

$$a_i = \frac{\rho_i^2}{\sigma_i^2} \tag{3.9}$$

$$b_i = \frac{\rho_i}{\sigma_i^2} \tag{3.10}$$

If a random variable ρ_i is distributed according to $\mathcal{G}(a_i, b_i)$, its inverse, $\delta_i = 1/\rho_i$, is distributed according to an Inverse Gamma, $\mathcal{G}^{-1}(a_i, b_i)$. Here a_i remains the shape parameter, but b_i is known as the scale parameter. The mean, μ_i and variance, Σ_i of δ_i are known and analytic, Equations (3.11) and (3.12).

$$\mu_i = \frac{b_i}{a_i - 1} \tag{3.11}$$

$$\Sigma_i = \frac{b_i^2}{(a_i - 1)^2(a_i - 2)} \tag{3.12}$$

Estimating $\rho_i \sim \mathcal{G}(a_i, b_i)$ is equivalent to estimating $\delta_i \sim \mathcal{G}^{-1}(a_i, b_i)$. This leads to a new formation for the predicted measurement given in Eqn. (3.13).

$$\bar{d}_i = \hat{s}\sqrt{\frac{\delta_i^2}{\hat{u}_m(3)^2} - \frac{2\delta_i}{\hat{u}_m(3)}\left(\hat{P}_i^m \cdot t_{m2o}^m\right) + (t_{m2o}^m \cdot t_{m2o}^m)} \tag{3.13}$$

We estimate δ_i with an EKF, so the partial derivative of Eqn (3.13) with respect to δ_i is required to find the variance of $\bar{b}_i$, Eqn (3.14).

$$\frac{\partial \bar{d}_i}{\partial \delta_i} = \frac{\hat{s}\left(\frac{2\delta_i}{\hat{u}_m(3)^2} - \frac{2}{\hat{u}_m(3)}\left(\hat{P}_i^m \cdot t_{m2o}^m\right)\right)}{\sqrt{\frac{\delta_i^2}{\hat{u}_m(3)^2} - \frac{2\delta_i}{\hat{u}_m(3)}\left(\hat{P}_i^m \cdot t_{m2o}^m\right) + (t_{m2o}^m \cdot t_{m2o}^m)}} \tag{3.14}$$

Now, an update to the mean and variance of δ_i can be found with the EKF equations.

$$\mu_i^+ = \mu_i^- + \frac{\left[\frac{\partial \bar{d}_i}{\partial \delta_i}\big|_{\mu_i^-}\right]^2 \Sigma_i^-}{\left[\frac{\partial \bar{d}_i}{\partial \delta_i}\big|_{\mu_i^-}\right]^2 \Sigma_i^- + \sigma_\omega^2}(d_i - \bar{d}_i) \tag{3.15}$$

$$\Sigma_i^+ = \Sigma_i^- - \frac{\left(\left[\frac{\partial \bar{d}_i}{\partial \delta_i}\big|_{\mu_i^-}\right]^2 \Sigma_i\right)^2}{\left[\frac{\partial \bar{d}_i}{\partial \delta_i}\big|_{\mu_i^-}\right]^2 \Sigma_i + \sigma_\omega^2} \tag{3.16}$$

The updated mean, μ_i^+, and variance, Σ_i^-, are then transformed back into Inverse Gamma shape and scale parameters. These are converted to shape and inverse scale parameters of a Gamma distribution representing the inverse depth, and then finally into the new values for the mean and variance of the estimate of inverse depth ρ_i.

In most situations, estimating δ_i with an EKF and Eqns. (3.15) and (3.16) leads to an estimator which is unbiased and consistent. An overview of the process is outlined below.

1. Transform LIDAR measurement into the map frame, continue if this measurement corresponds to a valid inverse depth estimate
2. Assume the inverse depth estimate belongs to a Gamma distribution, find its shape and inverse scale parameters
3. The depth now belongs to an Inverse Gamma distribution, find its shape and scale parameters
4. Use the shape and scale parameters to find the mean and variance of the Inverse Gamma distribution
5. Map the mean of the Inverse Gamma into a predicted measurement
6. Use the linearized mapping to find the partial derivative of the predicted measurement with respect to the depth, use this and the variance of the Inverse Gamma to find a predicted measurement variance

7. Perform an EKF update of the Inverse Gamma mean and variance using the LIDAR measurement, LIDAR measurement noise, predicted measurement, and predicted measurement variance

8. Map the updated Inverse Gamma mean and variance back into a Gamma mean and variance

9. Use the updated Gamma mean and variance for the new estimate of inverse depth

3.2.3.1 Map Point Initialization

Often a LIDAR measurement will be mapped into the keyframe and there will not be a valid inverse depth estimate at that pixel. In this case, the measurement forms the initialization of the inverse depth estimate. The inverse depth estimate is initialized with process similar to the one used to initialize the SLAM system. In this case, the new inverse depth estimates are scaled with the scale estimate found during keyframe creation. This is opposed to the scaling such that their mean is one as is done in SLAM initialization.

As before, we have a range measurement, d_i, which has been corrupted with zero-mean noise, ω. The variance of this noise is defined as σ_ω^2. This range measurement has been matched with a pixel whose inverse depth estimate we seek to initialize. We assume the distribution of our measurement is an Inverse Gamma, $\mathcal{G}^{-1}(a_i, b_i)$. The parameters a_i and b_i are defined as a function of the measurement and its variance according to Eqns (3.1) and (3.2).

The measurement can be converted to a 3D vector, P_i^l, with Eqn (3.24) and mapped into the coordinate system of the keyframe with Eqn (3.7). This set of operations is formalized in Eqn (3.17).

$$P_i^m = \frac{d_i}{\hat{s}} R_{o2m} P_i^l - t_{m2o}^m \tag{3.17}$$

We are interested in the third element (depth) of P_i^m. $P_i^m(3)$ has a distribution expressed as a transformation of the Inverse Gamma distribution of d_i with an offset of $-t_{m2o}^m(3)$. Thus its mean and variance can be found explictly as a function of d_i and σ_ω^2 as shown in Eqns (3.18) and (3.19).

$$\mu_i = \frac{d_i}{\hat{s}} \left(R_{o2m}(3,:) \cdot P_i^l \right) - t_{m2o}^m(3) \tag{3.18}$$

$$\sigma_i^2 = \frac{\sigma_\omega^2}{\hat{s}^2} \left(R_{o2m}(3,:) \cdot P_i^l \right)^2 \tag{3.19}$$

Where $R_{o2m}(3,:)$ is the 3^{rd} row of R_{o2m} and $(a \cdot b)$ is the inner product of a and b.

$P_i^m(3)$ is not distributed according to an Inverse Gamma for non-zero $t_{m2o}^m(3)$, however, we will assume that an Inverse Gamma is a decent approximation. This approximation is valid so long as μ_i is sufficiently large and σ_i^2 is sufficiently small. We can find the a_i and b_i parameters of this distribution according to Eqns (3.1) and (3.2).

We now assume our inverse depth estimate is distributed according to a Gamma with the same a_i and b_i parameters. The mean and variance of the inverse depth estimate can then be initialized according to Eqns (3.20) and (3.21).

$$\hat{\rho}_i = \frac{a_i}{b_i} = \frac{2 + \frac{\mu_i^2}{\sigma_i^2}}{\mu_i \left(1 + \frac{\mu_i^2}{\sigma_i^2}\right)} \tag{3.20}$$

$$\hat{\sigma}_i^2 = \frac{a_i}{b_i^2} = \frac{2 + \frac{\mu_i^2}{\sigma_i^2}}{\mu_i^2 \left(1 + \frac{\mu_i^2}{\sigma_i^2}\right)^2} \tag{3.21}$$

3.2.4 Map Smoothing

The map is smoothed using the LSD-SLAM smoothing algorithm after all EKF updates are performed and map points are initialized from the LIDAR measurements. Each inverse depth estimate is reassigned according to the average of the surrounding estimates weighted by the respective inverse estimate variances. Since the EKF update tends to significantly reduce the variance of an estimate, the surrounding inverse depths are pulled in the direction of the update by this process. Thus, the information from the LIDAR is able to update map points which are not directly measured. Smoothing can be justified from a theoretical perspective as the depths of adjacent pixels are highly correlated. Thus a LIDAR measurement of one pixel does provide information on the adjacent ones.

The algorithm used to perform the smoothing is identical to the one found in the unmodified LSD-SLAM.

3.3 SLAM Simulation

This thesis involved the creation of a simulation tool which produces RGB and LIDAR images for the SLAM system. This tool allows us to better understand the efficacy of our SLAM system when compared to experimental data by allowing the true environment and system pose to be known.

3.3.1 Simulated Environment

An example environment is shown in Figure 3.2. It contains a variety of cubes floating in space. The cubes have colored sides and black edges. The user is completely able to specify the location, size, and number of cubes. The software is able to render both interior and exterior faces of the cubes. This allows a large cube to encompass the scene and form the walls of the environment.

3.3.2 Simulation Outputs

The user provides a list of positions and orientations for sensor measurements to be generated at. These sensors are 480×640 RGB images, Figure 3.2, and 50×50 LIDAR images, Figure 3.3, which simulate the performance of a monocular RGB camera and flash LIDAR. The RGB images are generated according to a pinhole camera model [20]. Lidar range measurments are cre-

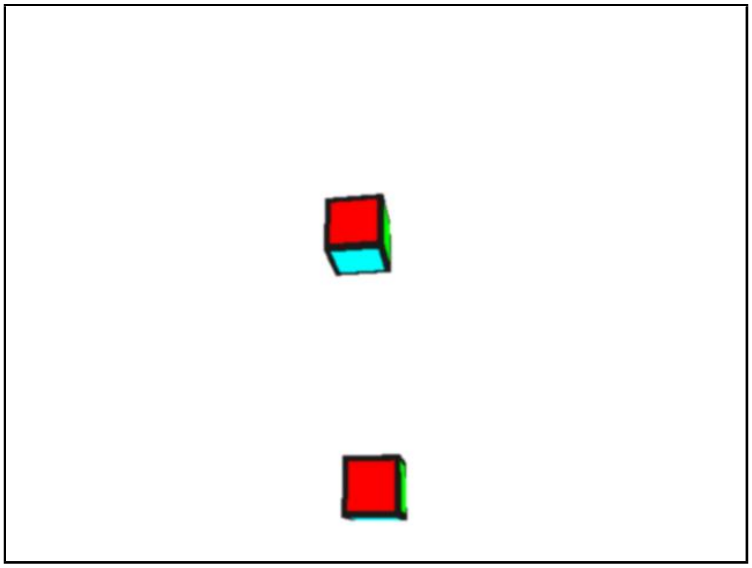

Figure 3.2: A RGB Image produced by the SLAM simulation. There are no simulation features in the background so the majority of the image is white. A black border has been added to the figure to denote the edge of the image, this is not present in the raw image file.

ated by measuring the range until intersection along a vector specified by an azimuth and elevation. The focal points and coordinate systems of the camera and LIDAR are assumed to be aligned, with each camera image corresponding to one LIDAR image taken at the exact same time.

A diagram of the pinhole camera model is shown in Figure 3.4.

3.3.3 Ray Tracing

Image generation is a computationally intensive process. Simulation speed was not a high priority so a naive ray tracing algorithm [**?**] was used over a more efficient option such as scanline rendering [**?**].

Inputs	Outputs
Sensor Position	RGB Image
Sensor Orientation	Lidar Image
Camera Calibration Matrix	
RGB and LIDAR Image Size	
Lidar Calibration Info	
Environment Features	

Table 3.1: The SLAM simulation inputs and outputs.

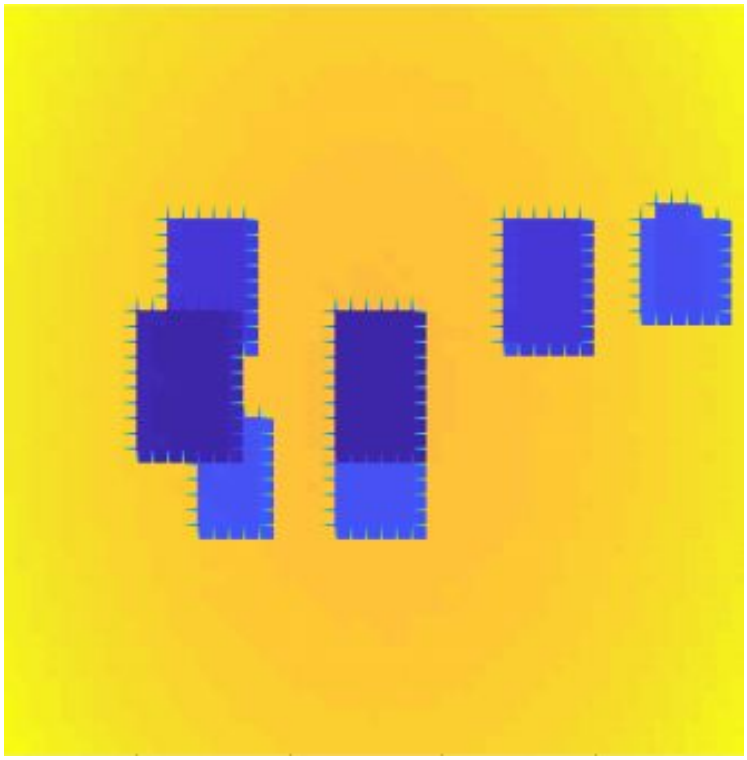

Figure 3.3: An example LIDAR image generated by the simulation. Note that LIDAR images are actually range measurements so false color has been added in order to simulate depth. The dashed lines at the edges of simulated objects show divisions in the pixels of the image. They are an artifact of the software used ot produce false color and are not represented in the actual images.

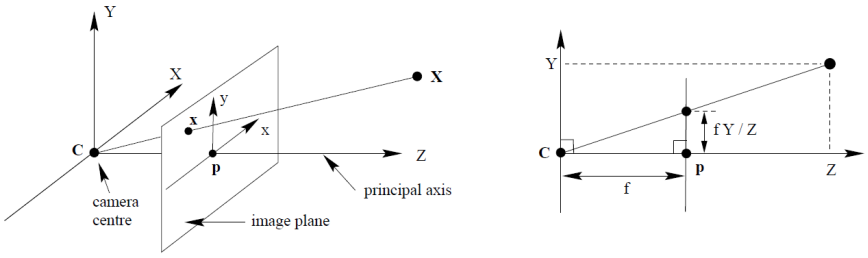

Figure 3.4: The pinhole camera model (Borrowed from [20]). This diagram shows how a 3D point X maps onto the image plane to point x.

The basic ray tracing process is to create a vector intersecting each pixel of an image. That vector is propagated forward through space until it intersects an object in the simulation. The object intersected determines the color value of the pixel. Consider a pixel with coordinates p_x and p_y. That pixel is used to form a homogeneous vector as in Eqn. (3.22).

$$p = \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \tag{3.22}$$

The vector p is then mapped through the inverse camera calibration matrix, K, to form a three dimensional vector in the map frame, P^m. K is defined as detailed in [20].

$$P^m = K^{-1}p \tag{3.23}$$

P^m is projected through the simulation environment and a check is performed with each object to determine if there is an intersection. All intersections as well as the distance to intersection are recorded. The intersection

with shortest distance is used to determine the color of the pixel. If no intersections are found, the pixel is assigned the color white.

After every pixel has been colored, a Gaussian blur is applied to the images to eliminate the aliasing found at the edges of simulation objects. An example of this aliasing is shown in Figure 3.5.

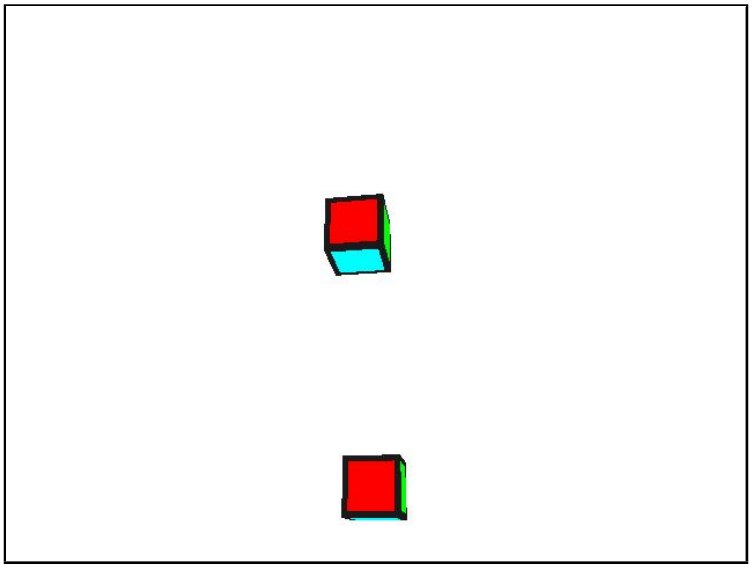

Figure 3.5: A RGB Image produced by the SLAM simulation without a Gaussian blur. Note the aliasing present in the form of jagged lines on the edges of the cubes. A black border has been added to the figure to denote the edge of the image, this is not present in the raw image file.

3.3.4 Lidar Image Generation

The distance to intersection found while generating RGB images is used to create LIDAR range images. Before this can be done, each LIDAR measurement must be associated with a pixel in the RGB images. Lidar measurements

are specified with an azimuth, θ, and elevation, ϕ angle. Azimuth is defined as a rotation about the LIDAR y-axis from the positive z-axis. Elevation is defined as a rotation about the LIDAR x-axis from the x-z plane. The azimuth rotation is applied first. This coordinate system is outlined in Figure 3.6. A given LIDAR measurement, d, can be converted to a 3D vector according to Eqn. (3.24).

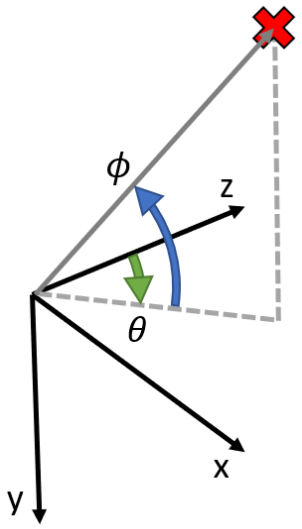

Figure 3.6: A diagram of the coordinate system defining a LIDAR measurement (red "X"). Note the definition of the blue elevation angle, ϕ, and the green azimuth angle, θ.

$$P^l = d \begin{bmatrix} -\sin(\theta)\cos(\phi) \\ \sin(\phi) \\ \cos(\phi)\cos(\theta) \end{bmatrix} \tag{3.24}$$

The vector P^l is associated with a pixel (p_x, p_y) with the relationship in Eqn. (3.25).

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = KP^l \rightarrow \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} x'/z' \\ y'/z' \end{bmatrix} \tag{3.25}$$

If the pixel pair (p_x, p_y) has an associated depth from RGB image generation, this depth is accepted as the range measurement for the LIDAR. If there was no intersection detected for this pixel, the range is simply recorded as zero. An example LIDAR image is shown in Figure 3.3.

The user can choose to corrupt the LIDAR measurements with gaussian noise for increased simulation realism.

Chapter 4

Results

The SLAM system created was validated through Monte Carlo testing. A trajectory was generated by varying some hyper-parameters. We did 1000 Monte Carlo runs generating 1000 sets of RBG and LIDAR images. These were fed into the SLAM system with LIDAR updates of the map enabled and disabled. In both cases, LIDAR images were still used to estimate the map scale at the keyframes, this was necessary in-order to produce a metric mapping and localization estimate.

4.1 Performance with a Rich Environment

This testing condition used 10 cubes in the center of the simulation environment as well as a encompassing box forming walls of the environment. This rich information environment allows for high-quality monocular SLAM tracking and mapping. Lidar serves to mostly improve the map. A sample image from this environment is shown in Figure 4.1.

The SLAM system initialized a map with the first RGB and LIDAR images. Once significant translation was detected, a new keyframe was formed with the next RGB image. The associated LIDAR image was then used to

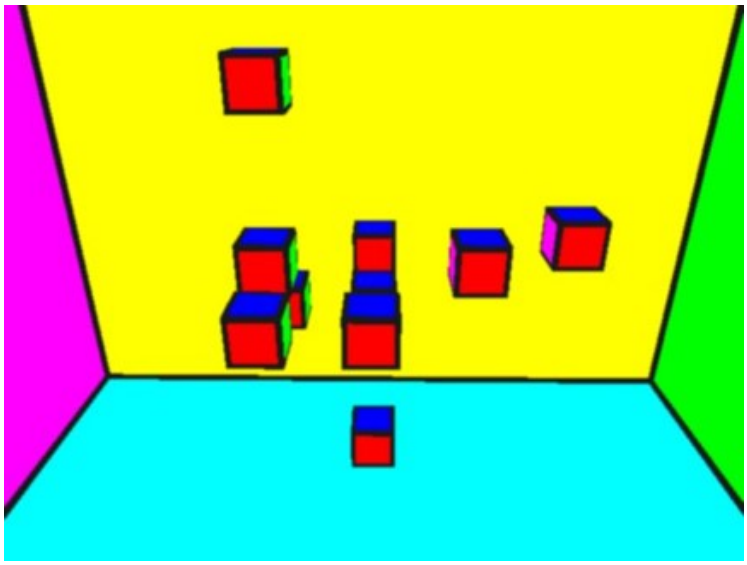

Figure 4.1: The rich simulation environment. Note the large number of objects as well as walls within the environment.

estimate scale. The map and pose estimates were saved before and after every subsequent operation until a new keyframe was formed. These estimates were compared to the truth for each Monte Carlo iteration. A variety of key metrics are outlined in the following sections.

One point to note is that the sensor's trajectory through each Monte Carlo run is stochastic. Keyframe selection and creation is determined by the estimate of these trajectories. One trajectory might result in 10 tracked frames being used to refine the keyframe while another uses 15. This means that the total number of tracked frames used varies across Monte Carlo runs. Figure 4.2 shows the number of Monte Carlo runs which are still refining the target keyframe at each tracked image.

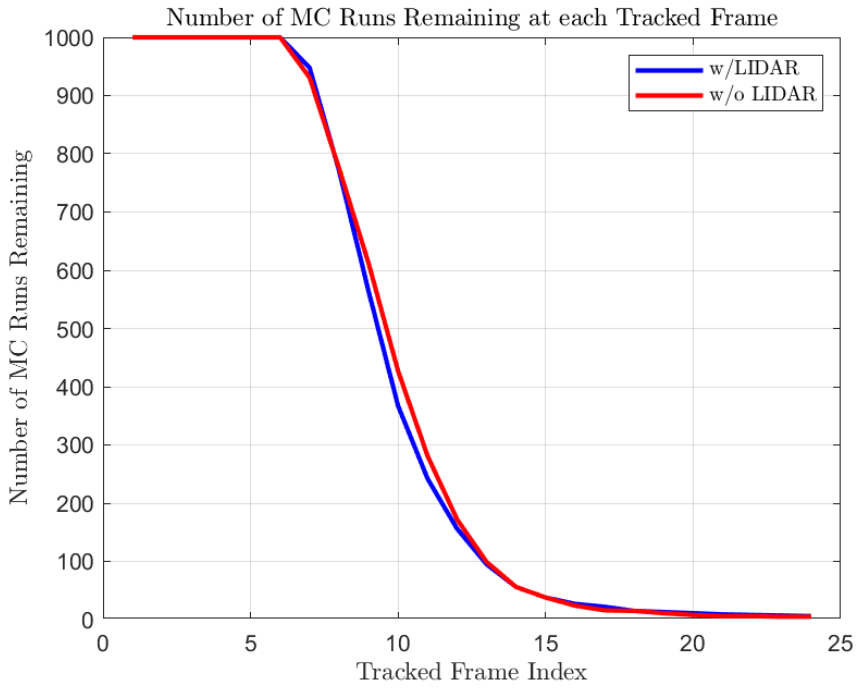

Figure 4.2: The number of Monte Carlo Runs remaining at each tracked frame with and without LIDAR.

The number of remaining runs begins to drop dramatically after approximately the 7^{th} tracked frame. Many metrics presented in this thesis are calculated at each frame by averaging across all Monte Carlo runs *remaining* at that point. For example, the average localization error will be calculated at both the 5_{th} and 10_{th} tracked frame. At the 5_{th} frame, there are 1000 data points used to calculate the average localization error since 1000 Monte Carlo runs remain. However, at the 10^{th} frame, there will likely only be 400 data points used since the SLAM system has decided to form a new keyframe prior to this point in 600 of the runs. Thus, the reader should focus their attention

primarily on statistics calculated before the 10_{th} tracked frame.

4.1.1 Scale Estimation Performance

The SLAM system estimates the scale of each local map upon keyframe creation. The true scale of the system is unknowable as every depth estimate contains error. One best estimate of scale, $\bar{s}_t$, can be found by looking at the localization estimates after each Monte Carlo run. The estimated distance translated from the keyframe at each RGB image can be compared to the true distance translated. Averaging over all localization estimates provides a scale estimate. This calculation is shown in Eqn (4.1).

$$\bar{s}_t = \sum_{i=1}^{N_t} \frac{||r_i^m||}{N||\hat{r}_i^m||} \tag{4.1}$$

A second best estimate of scale, $\bar{s}_m$, can be found by investigating every map point which has a depth associated after all map updates have been performed. This metric is calculated in the same manner as the system's estimate of scale, Eqn 3.6. However, for $\bar{s}_m$, all map points are considered, instead of only those with a LIDAR measurement.

$$\bar{s}_m = \frac{1}{N_m} \sum_{i=1}^{N_m} \frac{\hat{d}_i}{d_i} \tag{4.2}$$

Figure 4.3 compares $\bar{s}_t$ with $\hat{s}$. The sample covariance between the two estimates was calculated and used to create a covariance ellipse. This ellipse

along with the line $y = x$ were plotted to demonstrate that $\bar{s}_t$ and $\hat{s}$ are very well correlated.

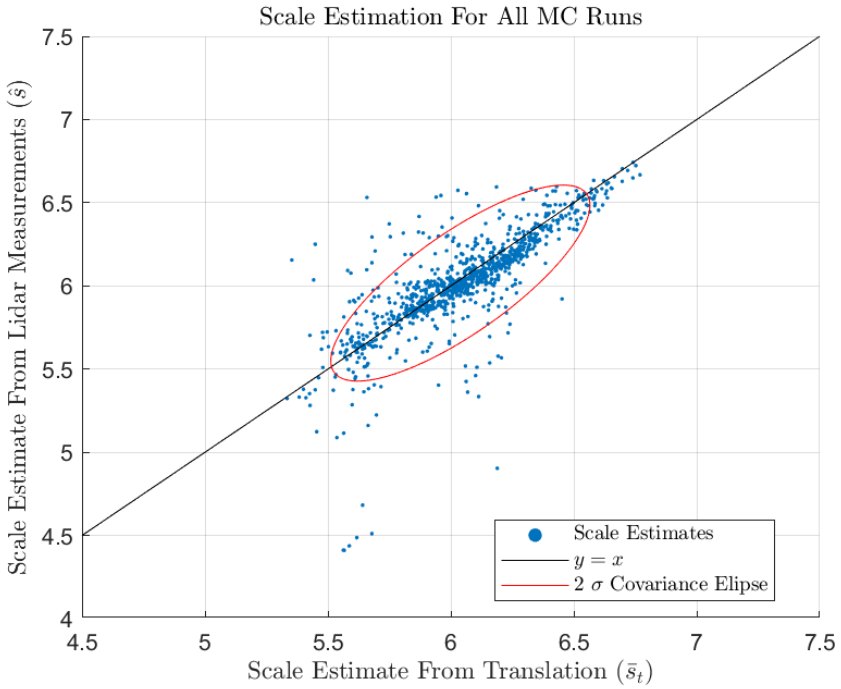

Figure 4.3: The scale estimation performance. For each Monte Carlo run $\bar{s}_t$ is plotted on the x-axis against $\hat{s}$ on the y-axis. The line $y = x$ and a covariance ellipse for $\bar{s}_t$ and $\hat{s}$ are also shown.

Figure 4.4 compares $\bar{s}_m$ with $\hat{s}$. The sample covariance between the two estimates was calculated and used to create a covariance ellipse. This ellipse along with the line $y = x$ were plotted to demonstrate that $\bar{s}_m$ and $\hat{s}$ are very well correlated.

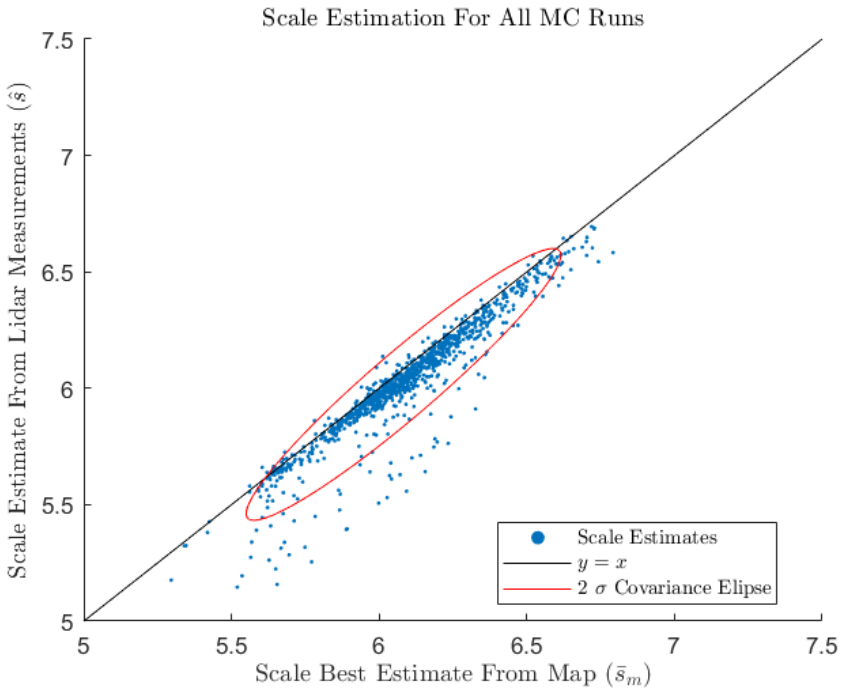

Figure 4.4: The scale estimation performance. For each Monte Carlo run $\bar{s}_m$ is plotted on the x-axis against $\hat{s}$ on the y-axis. The line $y = x$ and a covariance ellipse for $\bar{s}_m$ and $\hat{s}$ are also shown.

4.1.2 Localization Performance

The SLAM system provides localization estimates in the form of a rotation and translation from the keyframe to the most recent RGB image. Since absolute poses and rotations not observable in this SLAM system we choose to analyze the accuracy of these localization estimates with respect to the keyframe. Thus we compare the true translation from keyframe to the i-th frame r_i^m with its estimate, $\hat{r}_i^m$. Here the i denotes the image index while the m indicates the translation is expressed in the reference frame defined by the

keyframe. This reference frame is diagrammed in Figure 4.5

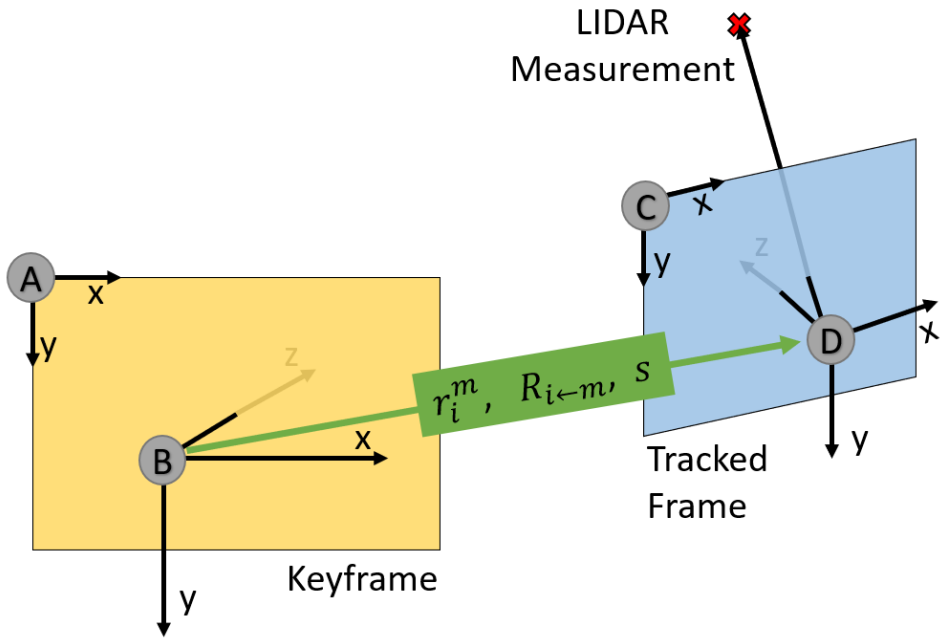

Figure 4.5: The coordinate systems for the RGB image forming the keyframe (A), non-metric local map (B), RGB image forming the i^{th} tracked frame (C), and metric body frame of the sensor during the i^{th} tracked image (D). Also labeled are the translation r_i^m, rotation $R_{i\to m}$, and scale s between the non-metric local map and metric body frame. A LIDAR measurement is shown in the metric body frame.

One additional factor for consideration is that the translation estimates are *non-metric*. Thus, we actually compare the true translation to the product of the estimated translation scaled by $\hat{s}$. This calculation is shown in Eqn (4.3). This is an imperfect metric as errors in mapping and localization cannot be fully separated from errors in scale estimation. Additional metrics can be created by comparing truth with the estimated translation scaled by $\bar{s}_t$ or $\bar{s}_m$ as shown in Eqns (4.4) and (4.5). These metrics are investigated after the

primary metric, $M_{4.3}$.

$$M_{4.3} = r_i^m - \hat{s}\hat{r}_i^m \tag{4.3}$$

$$M_{4.4} = r_i^m - \bar{s}_t\hat{r}_i^m \tag{4.4}$$

$$M_{4.5} = r_i^m - \bar{s}_m\hat{r}_i^m \tag{4.5}$$

The metric $M_{4.3}$ was calculated for each update of each Monte Carlo run. At each update, the mean and covariance of $M_{4.3}$ were calculated across all Monte Carlo runs. This was done for the case with and without LIDAR Measurements. Figure 4.6 displays the mean position error for each case. The Figure clearly shows the well known localization drift found in most SLAM systems. There is not a clear difference in drift between the cases until the 10^{th} tracked frame. At this point the system with LIDAR appears to show lower drift in the x and y axes. As described above, this is approximately the point when most of the SLAM systems in the Monte Carlo runs have decided to form a new keyframe. Further testing is required to determine if this deviation is due to reduced number of Monte Carlo runs used for analysis, or improved behavior due to LIDAR measurement incorporation.

The diagonal elements of the covariance matrix calculated from $M_{4.3}$ represent the error variance along each axis of the keyframe coordinate system. These are displayed in Figure 4.7. We immediately note a performance

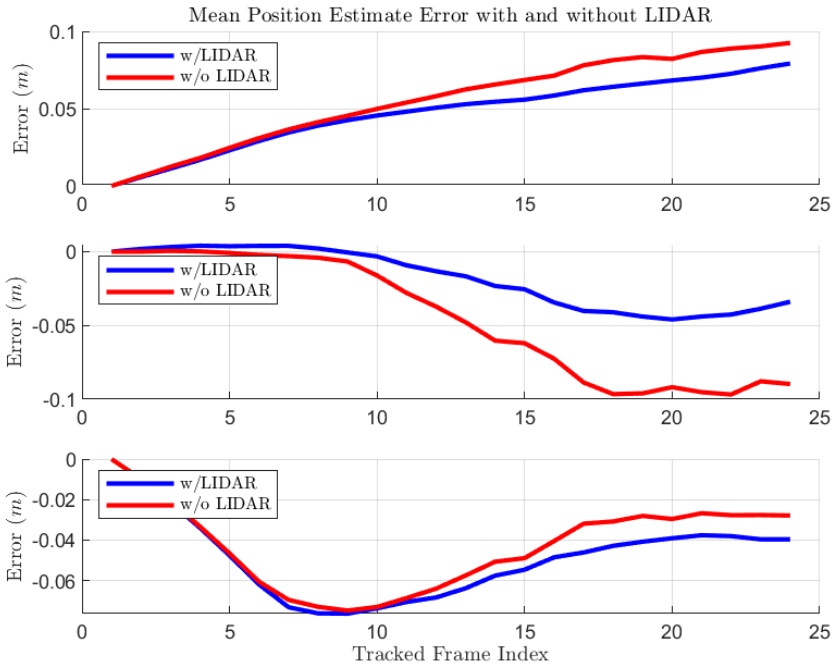

Figure 4.6: The mean position estimation error along each axis of the keyframe coordinate system, expressed in a metric representation.

difference between the systems with and without LIDAR. The variance in localization errors is lower in all cases when the system uses LIDAR. This is because LIDAR based map initialization and refinement result in a higher accuracy map. Localization is a direct function of the map, so a higher accuracy map translates directly to a higher accuracy localization estimate.

Figure 4.8 shows the positive and negative standard deviation of $M_{4.3}$ offset by the mean. This Figure was recreated with metrics $M_{4.4}$ and $M_{4.5}$ in Figures 4.9 and 4.10, respectively.

We also compare the true rotation between the coordinate frame defined

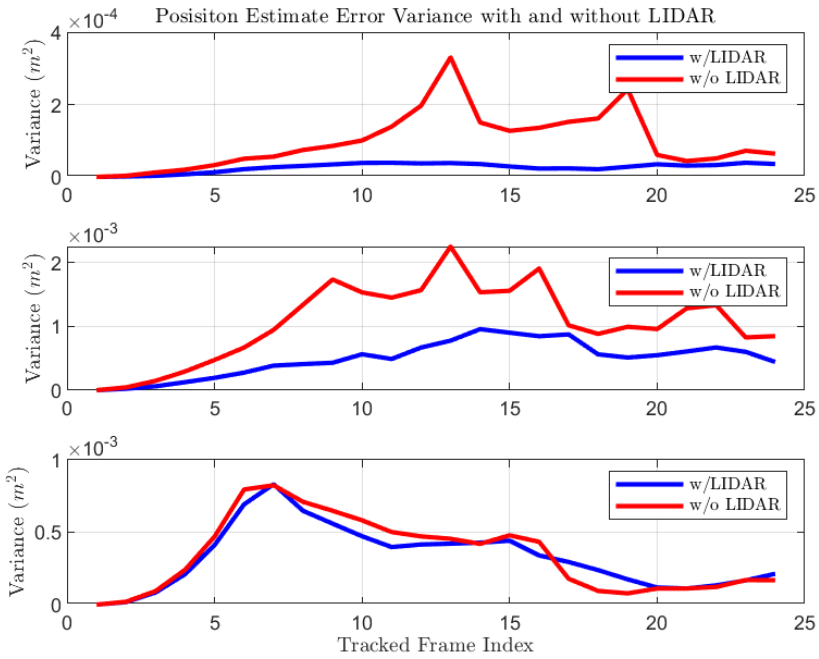

Figure 4.7: The variance of localization estimate errors, expressed in metric units in the keyframe coordinate system.

by the keyframe and the one defined by the i-th image, $R_{i\leftarrow m}$, with its estimate $\hat{R}_{i\leftarrow m}$. This is done through error Euler Angles. To find them, we first find an error rotation matrix, $R_e = R_{m\leftarrow i}\hat{R}_{i\leftarrow m}$. R_e is then converted to the error Euler Angles assuming an XYZ rotation order. This was done for the case with and without map updates from LIDAR. Figure 4.11 shows the average error Euler Angle for each axis. As with position, we see evidence of the drift characteristic in SLAM systems. The drift in the two systems roughly agrees until the 10^{th} tracked frame. After this point, the system's performance diverge from one another, but it is difficult to make meaningful conclusions

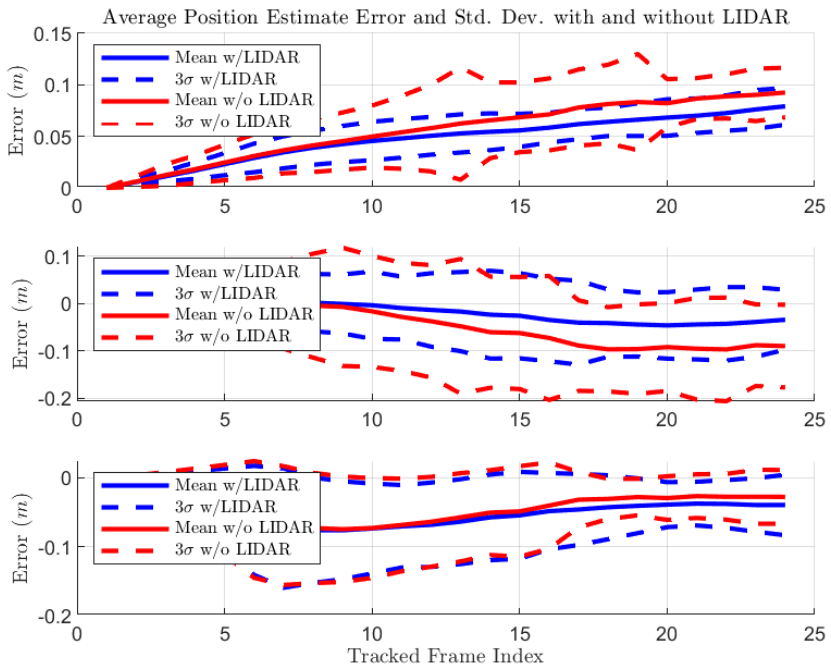

Figure 4.8: The mean and 3σ bounds for localization estimates

due to the attrition in Monte Carlo runs at this point.

The variance of each error angle was also calculated. This information is displayed in Figure 4.12. It is clear that using LIDAR results in a significantly improved error angle variance. This is again because an improved map results in improved localization efforts.

The mean and 3σ standard deviation of each Euler Angle is shown in Figure 4.13. It is made further clear from this Figure that LIDAR measurements significantly improve localization estimates.

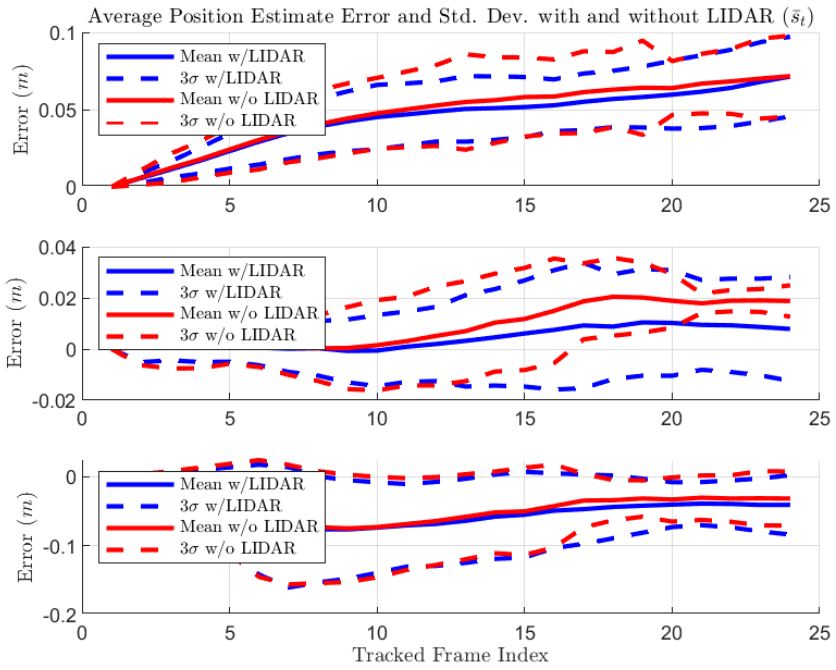

Figure 4.9: The mean and 3σ bounds for localization estimates using $\bar{s}_t$ as the scale estimate.

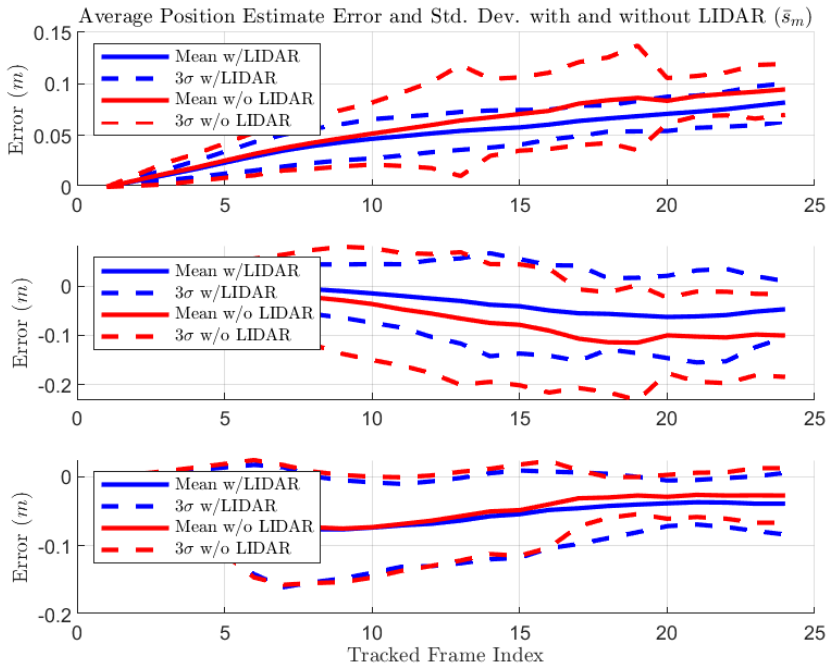

Figure 4.10: The mean and 3σ bounds for localization estimates using $\bar{s}_m$ as the scale estimate.

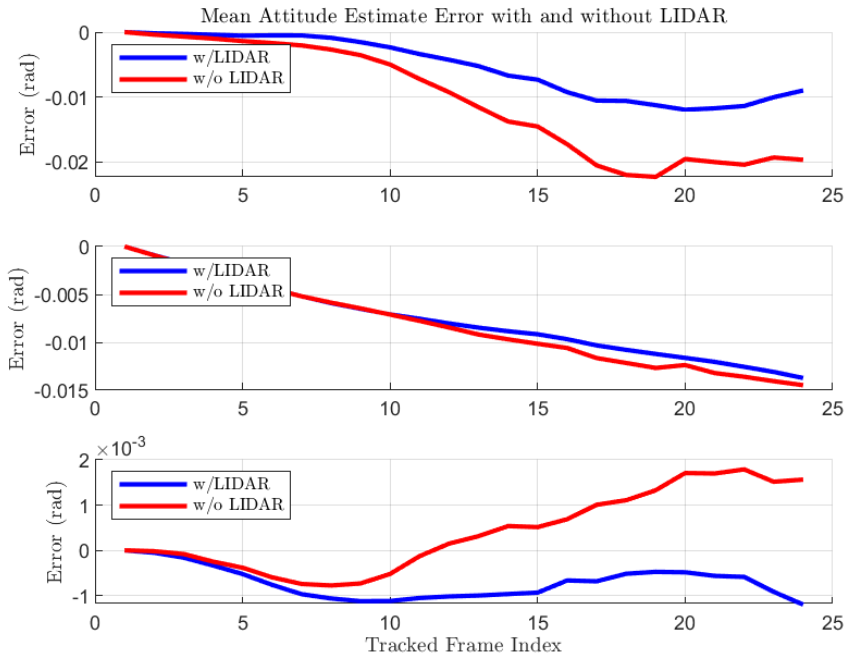

Figure 4.11: Attitude estimation error statistics with respect to the keyframe. Expressed in the local map frame.

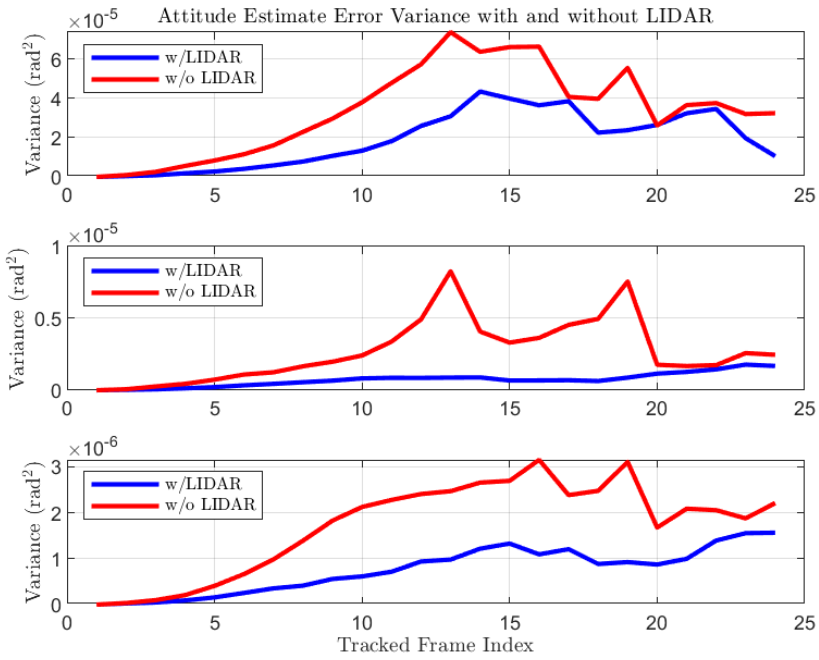

Figure 4.12: Attitude estimation error statistics with respect to the keyframe. Expressed in the local map frame.

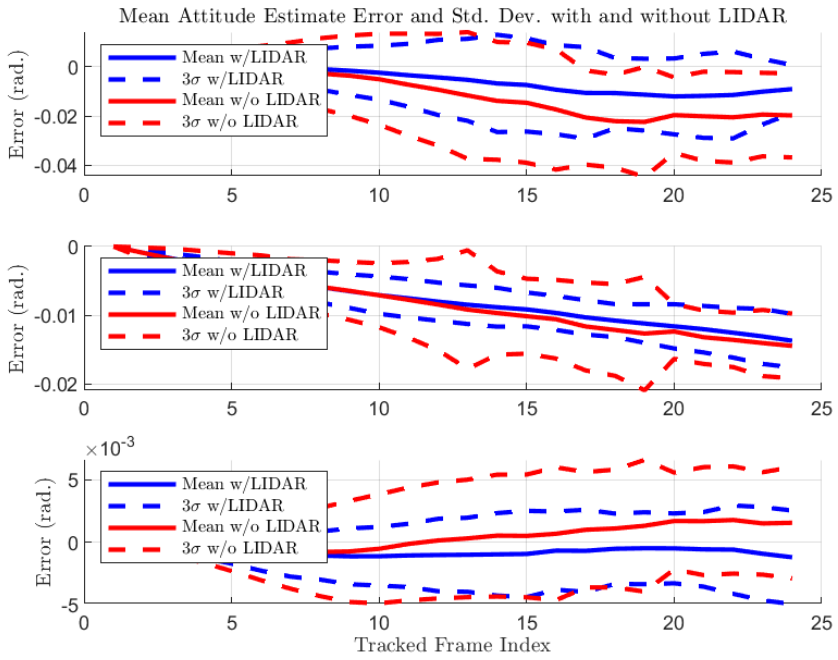

Figure 4.13: Attitude estimation error statistics with respect to the keyframe. Expressed in the local map frame.

4.1.3 Mapping Performance

The SLAM system provides mapping estimates as inverse depths, $\hat{\rho}_i$. For analysis we often choose to invert the estimates and provide metrics based on the depth estimate. A given local map consists of inverse depth estimates for some subset of the pixels of the keyframe. All images are created via simulation, thus the true depths for each pixel are known. The depth error, e_i, for a given pixel is calculated as shown in Eqn (4.6). As with the translation, the depth is stored in a non-metric representation. The scale estimate, $\hat{s}$, is used to provide the metric depth estimate.

$$e_i = \frac{1}{\rho_i} - \frac{\hat{s}}{\hat{\rho}_i} \tag{4.6}$$

Only a subset of the map points are directly updated by the LIDAR. We store the error before, e_i^-, and after, e_i^+, the EKF update for these points. Then we calculate the average reduction in error by the EKF across the updated points. This metric is shown in Eqn (4.7). Note that a positive $M_{4.7}$ indicates that on average, depth estimation error before the EKF update was *higher* than afterwards (*i.e.* the EKF improved depth estimation accuracy).

$$M_{4.7} = \frac{1}{N_{\text{EKF}}} \sum_{i=1}^{N_{\text{EKF}}} |e_i^+| - |e_i^-| \tag{4.7}$$

Figure 4.14 shows $M_{4.7}$ for each update of each Monte Carlo run. The data for each run is displayed with continuous lines for clarity. This should not be taken to indicate the system is behaving as a time series. In fact, the

movement of the camera between updates means that very few map points are updated by the EKF twice in a given Monte Carlo run.

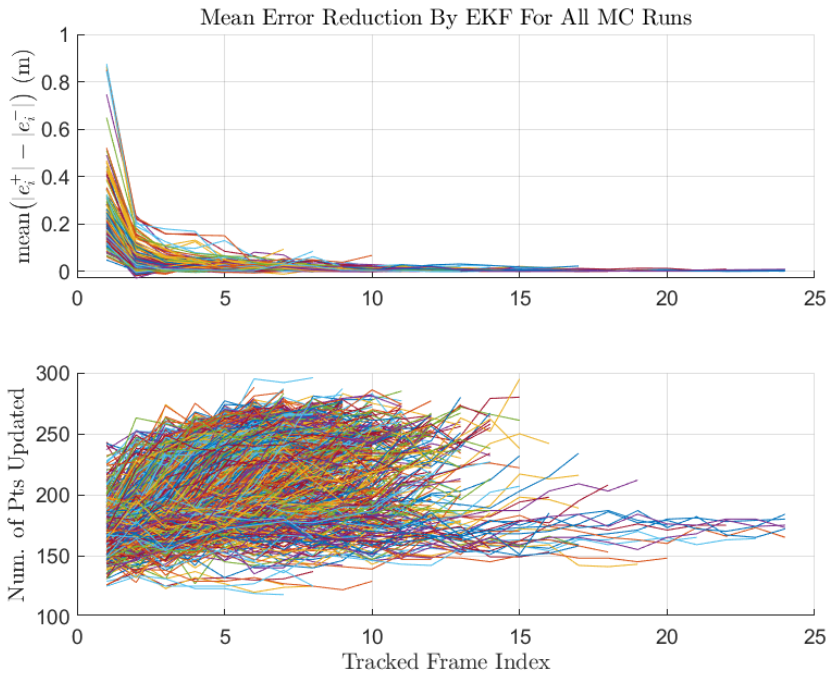

Figure 4.14: The mean error reduction by the LIDAR (top) and number of points updated via EKF (bottom). Error reduction is highest in the beginning when LIDAR provides the most information on the depth dimension. As the system translates away from the keyframe, less information is provided on the depth dimension, so error reduction tends to decrease.

EKF updates are only the first stage in LIDAR measurement incorporation in the SLAM system. The hole filling and smoothing procedures found in LSD-SLAM are applied after the updates are performed. The depth estimate error of all points affected by the smoothing process was stored before and after the operation. An average error reduction metric was calculated

from these points according to Eqn (4.8).

$$M_{4.8} = \frac{1}{N_{\text{smooth}}} \sum_{i=1}^{N_{\text{smooth}}} |e_i^+| - |e_i^-| \tag{4.8}$$

Figure 4.15 shows $M_{4.8}$ for each update of each Monte Carlo run. As in Figure 4.14, the data for each run is displayed with continuous lines for clarity. This should not be taken to indicate the system is behaving as a time series.

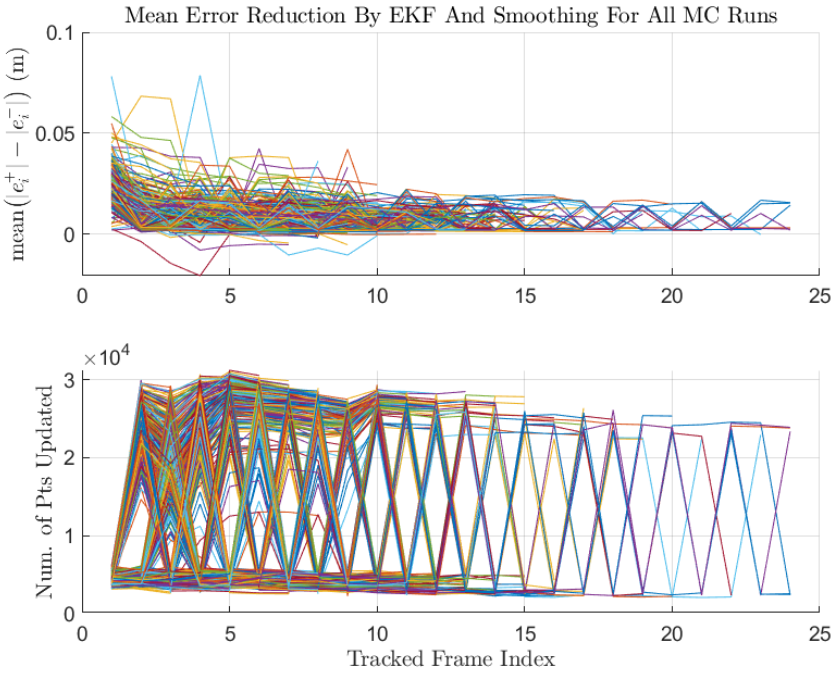

Figure 4.15: The mean error reduction by the LIDAR and smoothing algorithms (top) and number of points updated via EKF (bottom). Error reduction is highest in the beginning when LIDAR provides the most information on the depth dimension. As the system translates away from the keyframe, less information is provided on the depth dimension, so error reduction tends to decrease.

The LIDAR measurements were also used to initialize new map points. Figure 4.16 shows the average number of map points at each tracked frame. Also shown are the average number of map points which are directly initialized by LIDAR measurements. After keyframe initialization, the large majority of new points are initialized from LIDAR measurements. By the 24^{th} tracked frame, the average size has more than doubled from the inclusion of LIDAR points. Using LIDAR measurements clearly results in a significant improvement in mapping density.

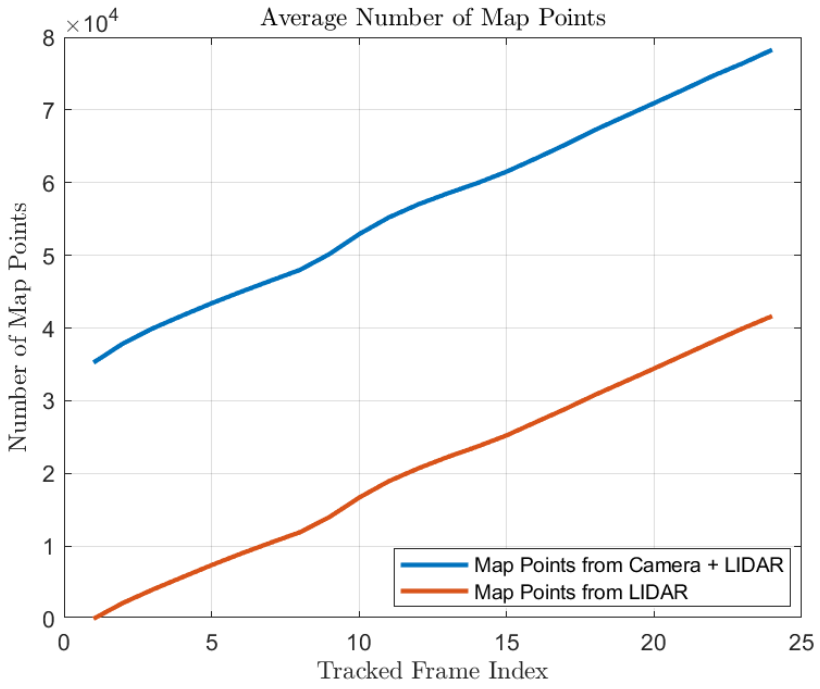

Figure 4.16: The average number of total map points compared to those which are initialized from LIDAR measurements alone. Note that LIDAR initialized points account for almost all new map points after keyframe creation.

Chapter 5

Conclusion

5.1 Research Summary

This thesis has presented a novel methodology for incorporating LIDAR range measurements into a direct monocular SLAM system. The open-source direct monocular SLAM system LSD-SLAM was used as a base for the development of this work. LSD-SLAM was modified to accept LIDAR images for three main purposes:

1. SLAM system initialization
2. Metric scale estimation
3. Map refinement

The first use of LIDAR measurements is for system initialization. The original LSD-SLAM algorithm chose to initialize the map randomly. This thesis was able to use calibration information to match LIDAR range measurements with map points. These map points were initialized according the LIDAR measurement and its associated variance. Clearly, a more accurate map is produced when initialization is done according to LIDAR measurements as opposed to randomly. The results of this thesis demonstrate that

this higher mapping performance then results in higher performance localization estimates.

The metric scale of the map is inherently unobservable in monocular SLAM. The second use of LIDAR images was in estimating this scale. The LIDAR image associated with each new keyframe was used to provide an estimate of the local map's scale. This scale estimate can be used to produce a metric map and localization measurements. The metric estimates are more useful for applications such as inspection or autonomous landing than the non-metric estimates produced by LSD-SLAM.

Finally, LIDAR images were used to refine the map. LIDAR measurements at each subsequent tracked RGB image after the keyframe were used to perform EKF updates on matched map points or to initialize new map points. In order to ensure the filter was consistent, the map estimates and range measurements were assumed to be distributed according to a Gamma and Inverse Gamma distribution respectively. This is in contrast to the traditional assumptions of Gaussian distributions in most EKF applications. The thesis was able to show that these EKF updates improved mapping performance on average, providing better mapping performance than LSD-SLAM.

5.2 Future Work

Future work can take a variety of directions. We have shown that LIDAR measurements can be incorporated via EKF into a non-linear least-squares monocular SLAM system. Future work could seek a more unified cou-

pling between the two measurement types. A SLAM system could be created which simultaneously uses direct monocular images and LIDAR measurements in a Bayesian filter. On the other hand, a future system could instead use both simultaneously in a larger non-linear least-squares problem.

Another direction for future work is the use of additional measurements. Inertial measurement units are some of the most common sensors in spaceflight and could provide significant benefits to SLAM localization performance. Inertial measurements are metric, so they also provide an additional avenue for scale estimation.

The final direction I will mention for future work is to expand the scope of research beyond SLAM. SLAM would never be performed for its own sake in spaceflight, the goal would always be to use the map and localization estimates to complete some additional task such as landing site selection. Future work could investigate these problems.

Bibliography

[1] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[2] Henning Lategahn, Andreas Geiger, and Bernd Kitt. Visual slam for autonomous ground vehicles. In *2011 IEEE International Conference on Robotics and Automation*, pages 1732–1737. IEEE, 2011.

[3] Jiang Bian, Xiaolong Hui, Xiaoguang Zhao, and Min Tan. A point-line-based slam framework for uav close proximity transmission tower inspection. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1016–1021. IEEE, 2018.

[4] Liang Zhang, Leqi Wei, Peiyi Shen, Wei Wei, Guangming Zhu, and Juan Song. Semantic slam based on object detection and improved octomap. *IEEE Access*, 6:75545–75559, 2018.

[5] Tao Yang, Peiqi Li, Huiming Zhang, Jing Li, and Zhi Li. Monocular vision slam-based uav autonomous landing in emergencies and unknown environments. *Electronics*, 7(5):73, 2018.

[6] Ji Zhang and Sanjiv Singh. Laser–visual–inertial odometry and mapping

with high robustness and low drift. *Journal of Field Robotics*, 35(8):1242–1264, 2018.

[7] Franz Andert, Nikolaus Ammann, and Bolko Maass. Lidar-aided camera feature tracking and visual slam for spacecraft low-orbit navigation and planetary landing. In *Advances in Aerospace Guidance, Navigation and Control*, pages 605–623. Springer, 2015.

[8] Alejo Concha, Giuseppe Loianno, Vijay Kumar, and Javier Civera. Visual-inertial direct slam. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1331–1338. IEEE, 2016.

[9] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[10] Jan Willem Marck, Ali Mohamoud, Eric vd Houwen, and Rob van Heijster. Indoor radar slam a radar application for vision and gps denied environments. In *2013 European Radar Conference*, pages 471–474. IEEE, 2013.

[11] Ling Chen, Aolei Yang, Huosheng Hu, and Wasif Naeem. Rbpf-msis: Toward rao-blackwellized particle filter slam for autonomous underwater vehicle with slow mechanical scanning imaging sonar. *IEEE Systems Journal*, 2019.

[12] Jack Brazzel, Fred Clark, and Zoran Milenkovic. Flash lidar based relative navigation. 2014.

[13] NASA. *Cygnus Cargo Spacecraft Docking With ISS*. Jul 2016.

[14] P Cheeseman, R Smith, and M Self. A stochastic map for uncertain spatial relationships. In *4th International Symposium on Robotic Research*, pages 467–474, 1987.

[15] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society, 2007.

[16] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[17] Thomas Schon, Fredrik Gustafsson, and P-J Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on signal processing*, 53(7):2279–2289, 2005.

[18] Michael Montemerlo and Sebastian Thrun. Fastslam 1.0. *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, pages 27–62, 2007.

[19] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[20] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[21] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Real-time monocular slam: Why filter? In *2010 IEEE International Conference on Robotics and Automation*, pages 2657–2664. IEEE, 2010.

[22] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[23] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[24] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[25] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[26] Luc Jaulin. Range-only slam with occupancy maps: A set-membership approach. *IEEE Transactions on Robotics*, 27(5):1004–1010, 2011.

[27] Emanuele Vespa, Nikolay Nikolov, Marius Grimm, Luigi Nardi, Paul HJ Kelly, and Stefan Leutenegger. Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151, 2018.

[28] Renato F Salas-Moreno, Ben Glocken, Paul HJ Kelly, and Andrew J Davison. Dense planar slam. In *2014 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 157–164. IEEE, 2014.

[29] Fabian Blochliger, Marius Fehr, Marcin Dymczyk, Thomas Schneider, and Rol Siegwart. Topomap: Topological mapping and navigation based on visual slam maps. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[30] Raúl Mur-Artal and Juan D Tardós. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017.

[31] Johannes Graeter, Alexander Wilczynski, and Martin Lauer. Limo: Lidar-monocular visual odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7872–7879. IEEE, 2018.

[32] Young-Sik Shin, Yeong Sang Park, and Ayoung Kim. Direct visual slam using sparse depth for camera-lidar system. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.